VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Skriptovatelný web crawler v C#

Scriptable Web Crawler in C#

2013

Jakub Gereg

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Jakub Gereg**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Skriptovatelny web crawler v C#
Scriptable Web Crawler in C#

Zásady pro vypracování:

Navrhněte a implementujte desktopovou aplikaci v jazyce C# určenou pro získávání dat z webu a jejich prezentaci.

Aplikace se bude skládat z:
1. Rozšíření skriptovacího jazyka pro popis postupu získávání dat.
2. Úložiště dat.
3. Jednoduchého web crawleru (stahujícího HTML stránky a obrázky na základě fronty generované uživatelským skriptem).
4. Uživatelského rozhraní pro prohlížení získaných dat.

Ke skriptování použijte vhodný existující skriptovací jazyk, který rozšíříte o API určené k manipulaci s webovými stránkami a ukládání získaných dat. Názorně předveďte toto API na skriptech (jeden pro textový obsah a jeden pro obrazový), které budou získávat data z některých známých služeb (např. zpravodajských webů, blogů, galerií, ...).

Seznam doporučené odborné literatury:

Podle pokynů vedoucího bakalářské práce.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Jakub Macek**

Datum zadání: 18.11.2011
Datum odevzdání: 07.05.2013

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Souhlasím se zveřejněním této bakalářské práce dle požadavků čl. 26, odst. 9 *Studijního a zkušebního řádu pro studium v bakalářských programech VŠB-TU Ostrava.*

V Ostravě 7.května 2013 .....

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 7.května 2013 .....

Rád bych na tomto místě poděkoval všem, kteří mi s prací pomohli. Mé poděkování patří především Ing. Jakubu Mackovi za ochotu a cenné rady při vedení mé bakalářské práce.

Abstrakt

Cílem této bakalářské práce je vytvořit skriptovatelný webový crawler. Základní myšlenka spočívá ve vytvoření aplikace, která bude automatizovaně procházet webové stránky v závislosti na skriptu, jenž předem specifikován samotným uživatelem. Před samotnou implementací bylo zapotřebí analyzovat vhodný skriptovací jazyk, který by zastupoval použití a obsluhu crawleru. Dále bylo třeba nastudovat základní pravidla pro průchod crawleru webem spolu s omezením těchto pravidel. Výsledkem mé práce je funkční aplikace umožnující zjednodušené prohledávaní informací na internetu a jejich následnou interpretaci.

Klíčová slova: URL, HTTP, C#, crawler, Web crawler, skriptovací jazyk

Abstract

The aim of this bachelor thesis is to create a scriptable Web crawler. The basic idea was to create an application that would automatically browse web pages based on user-specified script. However, we needed to analyze which scripting language would be appropriate for operating with crawler and it was also necessary to learn the basic rules for the movement of the web crawler, even before we began. Also, we needed to take in account some limitations, that come with crawlers itself. In result, there is a functional appliaction, allowing simplified searches on the Internet.

Keywords: URL, HTTP, C#, crawler, Web crawler, scripting language

Seznam použitých zkratek a symbolů

DLR	–	Dynamic Language Runtime
CLR	–	Common Language Runtime
API	–	Application programming interface
SQL	–	Structured Query Language
GUI	–	Graphical user interface
MIT	–	Massachusetts Institute of Technology
OS	–	Operating system
URL	–	Uniform Resource Locator
HTTP	–	Hypertext Transfer Protocol
HTML	–	Hyper Text Markup Language
OOP	–	Object-oriented programming
DLL	–	Dynamic-link library

Obsah

Seznam obrázků

Seznam výpisů zdrojového kódu

1 Úvod

V dnešní době se prakticky veškeré informační zdroje nacházejí na internetové síti, a proto je těžké si představit vyhledávání informací stránku po stránce. Této úlohy se ujaly automatizované služby, které informační zdroje vyhledávají na základě prohlížení celé internetové sítě. Služby vznikly zároveň s nástupem internetu, a to v 90. letech. V roce 1994 byla odstartována éra vyhledávačů „první katalog Yahoo! Directory“. V následujících letech začali vznikat další vyhledávače, které známe dnes. V počítačové terminologii se místo pojmu „vyhledávač“, používají pojmy jako je „crawler“, „robot“ nebo „spider“ tzv. česky „pavouk“.

Práce tedy vyžadovala teoretickou studii crawlerů a jejich chování na internetu, které je podmíněno určitými pravidly. Na základě prostudování různých faktů jsem vytvořil vlastní webový crawler. Tento crawler umí stahovat a prezentovat data, pomocí skriptů, které si uživatel může napsat sám podle potřeby.

Pro nastínění využitelnosti uvedu např. blog prof. Ing. Ivo Vondráka, CSc. rektora VŠB, kde pomocí jednoduchého skriptu, který je uveden v příloze, jsou k dispozici vždy aktuální informace o událostech na škole i mimo ní.

Výhodou programu je možnost prohlížení informací v režimu offline, například na cestách, dovolené apod., což je velmi užitečné v případě, že není k dispozici připojení k internetu.

2 Web crawler

2.1 Co je to Web crawler

Pod pojmem Web crawler, neboli webový crawler, si představíme automatizovaný počítačový program, který prochází webový obsah na internetu, dle vymezených kritérií. Crawlery jsou také využívány vyhledávacími službami pomocí vyhledavačů, u kterých zastávají funkci aktualizace dat.

Takovéto procházení internetem a indexovaní je většinou označováno jako crawling.

2.2 Jednoduchý crawler

Nejjednodušší procházení webového obsahu pracuje na principu načtení seznamu stránek (resp. URL adres), které stáhne a vybere z nich jen URL odkazy. Ty poté zařadí do fronty a pokračuje stejným zpusobem.

Složitější crawlery například řadí do fronty přednostně ty odkazy, s vyšším hodnocením, které počítají podle vlastních metodik např. (**Google** – PageRank, **Seznam** - S-Rank).

2.3 Vlastní crawler

Tato aplikace bude nabízet řešení situací, kdy je potřeba shromáždit dané informace bez nutností prohlížení webových stránek. Pomocí této aplikace není využito manuální vyhledávání stránek uživatelem, ale vyhledávání automatické. Díky automatu tak uživatel ušetří čas, který by trávil vyhledáváním konkrétního typu informace, např. blogy, návody, obrázky a jiné. Program za uživatele vybere pouze požadované informace, které zobrazí. Vybrané informace jsou již bez reklam, anket apod.

Takový postup se vyplatí u rozsáhlých webů, kde je složitější vyhledat konkrétní informaci.

3 Analýza vhodných skriptovacích jazyků

Pro vhodný skriptovací jazyk je potřeba nejprve vyhledat některá z řešení. Vybral jsem si tedy takové možné řešení, které je reálné jak pro mě, tak i pro výsledný program. V jednotlivých jazycích se zabýváme syntaxí a způsobem volaní metod. Volání metod dále rozdělujeme na:

- **Interní** – metody napsané přímo ve skriptu.
- **Externí** - metody, které jsou definovány v programu a skript je pouze spouští.

3.1 IronPython

Název sám o sobě již napovídá, že se jedná o implementaci jazyka Python pro platformu .NET.

Python jako takový, je objektově orientovaný.[3] Historie tohoto jazyka spadá do roku 1991, což z něj činí relativně mladý jazyk. Patří mezi hybridní jazyky, což znamená, že se jedná o programovací a zároveň o skriptovací jazyk.

K jeho hlavní výhodě, pro některé možná i přítěží, je jeho snaha naučit programátora psát čitelný a "pěkný kód". Tato snaha spočívá v psaní pomocí odsazování tabulátorem.

3.1.1 Ukázka syntaxe

```
def fce(a,b):
  return a+b

found = False
abc = array([7,2,4,5,6])

for i in range(0,4):
  if abc[ i ]; > 5:
     found = True
```

Výpis 1: Ukázka syntaxe v Python

3.1.2 Ukázka IronPython v C#

K propojení IronPython s C# [4] aplikací je zapotřebí dll knihovny, které jsou k dispozici na internetových stránkách `http://ironpython.net/`.

Ze stažených souborů nás především zajímají tyto:

- IronPython.dll
- IronPython.Modules.dll
- Microsoft.Dynamic.dll
- Microsoft.Scripting.Debugging.dll
- Microsoft.Scripting.dll

Když budou tyto knihovny přidány do projektu Visual Studia, přes References->Add Reference. Můžeme začít spouštět Python skripty například takto:

```
using IronPython.Hosting;
using Microsoft.Scripting;
using Microsoft.Scripting.Hosting;
...
ScriptEngine pyEng = Python.CreateEngine();
var file = File.ReadAllText(@"C:\ScriptTest\python.py");

ScriptSource source = pyEng.CreateScriptSourceFromString(file,SourceCodeKind
    .Statements);
var myFunction = new Action<string>(delegate(string text){ Console.WriteLine
    (text); });

ScriptScope scope = pyEng.CreateScope();
scope.SetVariable("csharpfce", myFunction);

source.Execute(scope);
var classA = scope.GetVariable("A");
var instance = scope.Engine.Operations.CreateInstance(classA);

var fce = pyEng.Operations.GetMember<Func<int,int,int>>(instance,"fce");

var result = fce(1,2);

Console.WriteLine(result);
```

Výpis 2: Propojení C# a IronPythonu

3.1.3 Zhodnocení

Toto řešení se mi jeví jako nevhodné pro tento typ problému. Syntaxe vyžaduje znalost jazyka Python, minimálně v odsazování pomocí tabulátorů, což může nezkušeným uživatelům činit problémy. Registrace vlastních funkcí funguje pomocí delegátů, což podle mého nepatří mezi nejlepší řešení. Jakožto nejlepší řešení si představuji implementaci pomocí vlastních C# metod, které zaregistruji. Ve skriptu pak tyto metody zavolám pod definovanými názvy.

3.2 IronRuby

Jedná se o další skriptovací jazyk z rodiny „Iron“, který společnost Microsoft vydala jako open-source a zároveň jeho vývoj přenechala na komunitě.

Jedná se o jazyk Ruby a jeho integaci do prostředí .NET. Ruby je plně objektově orientovaný jazyk, což znamená, že co je v Ruby, je objekt.

Jazyk byl vyvinut v roce 1993, ale i přesto je ve světě programování velmi známý a to díky svojí jednoduchosti a přehlednosti kódu.

Stejně jako IronPython je IronRuby implementován nad DLR knihovnou.

3.2.1 Ukázka syntaxe

```
def fce(a, b):
   return a+b
end

found = false
abc = [7, 2, 4, 5, 6]
  for i in 1..5
    if (abc[ i ]>5)
      found = true
    end
  end
```

Výpis 3: Ukázka syntaxe v Ruby

3.2.2 Ukázka IronRuby v C#

Pro spuštění IronRuby v jazyku C#, je zapotřebí stáhnutí konkrétní knihovny z adresy `http://www.ironruby.net/` . Jedná se především o knihovny:

- IronRuby.dll
- IronRuby.Libraries.dll
- IronRuby.Libraries.Yaml.dll

- Microsoft.Dynamic.dll
- Microsoft.Scripting.dll
- Microsoft.Scripting.Core.dll

Knihovny se připojí jako reference k projektu a je možné spouštět Ruby skripty. Je důležité napsat program pro spuštění skriptu ruby.rb, který je uveden v příloze.

```
using IronRuby;
...
var runtime = Ruby.CreateRuntime();
var engine = runtime.GetEngine("rb");

var file = File.ReadAllText(@"C:\ScriptTest\ruby.rb");

ScriptSource source = engine.CreateScriptSourceFromString(file,
    SourceCodeKind.Statements);
var scope = engine.CreateScope();
scope.SetVariable("csharpfce",new MyClass());

source.Execute(scope);

var cls = engine.Runtime.Globals.GetVariable("A");
var instance = engine.Operations.CreateInstance(cls);
object[] args = { 1, 2 };
var result = engine.Operations.InvokeMember(instance, "fce", args);

Console.WriteLine(result);
...
public class MyClass
{
  public void MyFunction(String text)
  {
    Console.WriteLine(text);
  }
}
```

Výpis 4: Propojení C# a IronRuby

3.2.3 Zhodnocení

K tomuto jazyku je můj postoj neutrální. Syntaxe je přehledná, rychlost spouštění skriptu relativně přijatelná. Přítěží však je, že funkci jako takovou, nelze spustit bez vytvoření instance třídy, která ji obsahuje. Tento problém se nachází také v IronPython.

3.3 Javascript.NET

Ač by se mohlo podle názvu zdát, že se jedná o integraci JavaScriptu do .NET, tak tomu tak není. Toto řešení integruje "Google's V8 Javascript engine", což je open source Javascriptový engine od firmy Google, který je napsán v jazyce C++ a je použit v prohlížeči Google Chrome.

3.3.1 Ukázka syntaxe

```
function fce(a,b)
{
  return a+b;
}

var found = false;
var abc = [7, 2, 4, 5, 6];
for (i=0; i<5; i++)
{
  if (abc[ i ]>5)
    found = true;
}
```

Výpis 5: Ukázka syntaxe v Javascript

3.3.2 Ukázka Javascript.NET v C#

Na stránkách `http://javascriptdotnet.codeplex.com/` , je potřeba stáhnout .zip archív s dll knihovnami.

V rozbalené složce se nachází knihovny, jak pro Framework 4.0, tak i pro starší verzi 3.5. Stejné rozdělení je i pro x86 a x64 typ procesoru. Knihovna, kterou je potřeba importovat do projektu:

- Noesis.Javascript.dll

Nyní si předvedeme jednoduché volaní javascript funkce, která se nachází také v příloze s názvem „javascript.js“.

```
using Noesis.Javascript;
...
var context = new JavascriptContext();
context.SetParameter("csharpfce",this);

var script = File.ReadAllText(@"C:\ScriptTest\javascript.js");
context.Run(script);

Console.WriteLine(context.Run("fce(2,1);"));
...
public void MyFunction(string text)
{
  Console.WriteLine(text);
}
```

Výpis 6: Propojení C# a Javascript.NET

3.3.3 Zhodnocení

Javascript, jako skriptovací jazyk, má poněkud složitější syntax. Je potřeba psát ukončující znaky, začátky a konce bloků pomocí závorek. Mimo jíné má javascript prototypové OOP, tzn. že nepoužívá klasickou třída-instance koncepci.Dle mého názoru, je toto řešení vhodnější pro zkušenější uživatele.

Na druhou stranu, deklarace a spouštění metod je jednoduché a efektivní. Pro spuštění metody přímo ze skriptu stačí zavolat **context.Run(„fce(2,1)“)**, kde metoda Run vrací výsledek typu objekt. Abychom mohli vytvořit vlastní „Externí“ metodu umístěnou v aplikaci, musíme jí nejprve zaregistrovat. Registrace probíhá pomocí příkazu **SetParameter(”csharpfce”,this)**, kde první parametr je název pod jakým bude metoda volána

ze skriptu, a this je objekt. Například libovolná jiná třída. Tato třída může obsahovat implementované metody, které pak jednoduše můžeme spouštět pomocí zaregistrovaného názvu. Například ve tvaru **csharpfce.MyFunction("hello from javascript.net")**.

3.4 DynamicLua

DynamicLua je nástavbou nad knihovnou LuaInterface, která propojuje jazyk Lua a .NET. CLR.DynamicLua navíc umožňuje využívání nových vlastností nejnovějšího Frameworku .NET 4.0.

Jazyk Lua je rychlý, přehledný a klade důraz hlavně na vysoký výkon. Mezi negativní vlastnosti patří např. špatné zpracovávání chybových hlášení, ve kterých nelze dohledat daný řádek s chybou.

3.4.1 Ukázka syntaxe

```
function fce(a,b)
  return a+b
end

found = false
abc = {7, 2, 4, 5, 6}
for i=0,4,1 do
  if abc[ i ] > 5 then
    found = true
  end
end
```

Výpis 7: Ukázka syntaxe v Lua

3.4.2 Ukázka DynamicLua v C#

DynamicLua lze stáhnout z internetové stránky `http://dynamiclua.codeplex.com/` . Tento archív obsahuje 3 důležité knihovny:

- lua.dll
- luaInterface.dll
- DynamicLua.dll

Pro naše účely stačí do projektu importovat první dvě uvedené knihovny. DynamicLua pouze zjednodušuje obsluhu LuaInterface.

```
using LuaInterface;
...
var lua = new Lua();

lua.RegisterFunction("csharpfce", this, this.GetType().GetMethod("MyFunction
    "));

lua.DoFile(@"C:\ScriptTest\lua.lua");
lua.DoString("result = fce(1,2)");

var res = lua["result"];

Console.WriteLine(res);
...
public void MyFunction(string text)
{
  Console.WriteLine(text);
}
```

Výpis 8: Propojení C# a DynamicLua

3.4.3 Zhodnocení

Lua z hlediska syntaxe je jednoduchá na naučení a orientaci v kódu. V porovnání s ostatními skriptovacími jazyky, se registrace vlastních funkcí a jejich následovné volání implementovala poněkud snáz. Stačí napsat **RegisterFunction(”jmenofunkce”,this,...)**, posledním parametrem je reflexe na danou metodu, která se má spustit. Tuto metodu získáme pomocí příkazu **GetType().GetMethod(”nejakafunkce”)**. Pro spuštění funkce „**nejakafunkce()**“ ze skriptu stačí zavolat “**jmenofunkce()**“. Základním nedostatkem je fakt, že jazyk Lua nepodporuje enumerátory. Například foreach cykly nad všemi objekty v kolekci. Tento problém, je však už mnohokrát řešený na internetu, proto jeho řešení nebylo složité najít.

3.5 Závěrečné shrnutí

Při závěrečné analýze skriptovacích jazyků jsem si zvolil jazyk Lua, který mne zaujal svou jednoduchostí při implementaci do C#. Ze všech zmíněných vykazoval nejjednodušší práci pro uživatele, který vytváří a pracuje se skripty. Jako alternativu bych volil ještě Javascript.NET, který jsem si nevybral, jelikož má složitější syntaxi.

4 Skriptovací Jazyk

4.1 O Jazyku Lua

Jazyk Lua byl vytvořen v roce 1993 v Brazílii. Aktuální dostupná verze je Lua 5.2 a byla oficiálně vydána 16. prosince 2011.

Lua je jednoduchá, rychlá a hlavně volně dostupná jako open-source pod licencí MIT.

Lua je populární především pro její využití v mnoha herních aplikací, jako je například World of Warcraft, MDK2 a mnoha dalších, čímž vytváří rozšířitelný prostor pro mnohé vývojáře.

4.2 Vlastní API

Pro manipulaci s crawlerem bylo nutné navrhnout vlastní API, jejichž základní funkce a principy jsou popsány v této kapitole.

4.2.1 Funkce

- **enum**(*object*)[1] - nahrazuje foreach cyklus, prochází celou kolekci dat postupně. Příklad: for one in **enum**(collection) do
- **HttpGet**(*string*) - funkce stáhne webovou stránku z dané URL a předá ji jako objekt WebPage.
- **GetAllPagesUnder**(*WebPage*) - funkce přijímá objekt typu WebPage (startovní url adresu) a vrací kolekci všech nalezených zpětných odkazů. Zpětné odkazy hledá pouze v rámci domény. Kolekce je typu Uri.
- **GetAllLocalPagesUnder**(*WebPage*) - funkce je podobná jako GetAllPagesUnder, jediný rozdíl je ve vrácené kolekci zpětných odkazů. Tato metoda zužuje výběr odkazů, na odkazy dané složky a jejich podsložek. Např. zadáme-li ”http://vsb.cz/slozka” ve výsledku nebude žádný odkaz směřující mimo tuto složku např. ”http://vsb.cz/jinaslozka”.

[1] Lua nepodporuje procházení kolekcí, tak jako jej umožnuje cyklus *foreach*. Abychom je mohly používat musí být definována metoda enum.

- **Save**(*params object[]*) - funkce přijímá libovolný počet parametrů typu object. Tyto hodnoty ukládá do kolekce a čeká na volání metody Transaction(). Parametry musí být ve tvaru (”název sloupce 1”,hodnota sloupce 1,”název sloupce 2”,hodnota sloupce 2,...)
- **SaveTo**(*string,string,params object[]*) - funkce je stejná jako Save, s tím rozdílem že první 2 parametry definují cestu k databázovému souboru, a jméno tabulky do které se bude ukládat.Tzn. ignoruje nastavení v globální proměnné DATABASE.
- **Transaction**() - funkce pomocí jedné SQL transakce, spustí všechny dotazy uložené pomocí metody Save.
- **AddTable**(*object*) - tato funkce vytváří tabulku do databáze.V konstruktoru přijímá parametr typu object. Zde se uvádí SQL dotaz pro vytvoření tabulky ”CREATE TABLE ..”, nebo je možnost předat pomocný objekt typu Table.
- **TableName**(*string*) - pomocná funkce pro tvorbu ”CREATE” SQL příkazu. Vrací objekt typu Table, na který můžeme zavolat další metody, jako např. AddAttribute.
- **CombinePath**(*string,string*) - tato funkce zkombinuje základní cestu s relativní a vytvoří z ní absolutní cestu. Např. ”C://zakladni” a ”./relativni/cesta” bude ve výsledku ”C://zakladni/relativni/cesta”.

4.2.2 Datové typy/objekty

- **WebPage**(*string* nebo *Uri*)- Načítá ”parsuje” webovou stránku podle zadaného url.
  - **Url** - vrací Url adresu jako objekt typu Uri.
  - **Encoding** - obsahuje kódování stránky, objekt typu Encoding.
  - **InnerHtml** - vrací HTML kód.
  - **Xpath**(*string*) - podle XPath dotazu vrací kolekci elementů typu HtmlElement.
- **HtmlElement**(*HtmlNode*) - třída, která uchovává informace o html elementu/tagu.
  - **InnerText** - obsahuje text, který je v daném elementu obsažen.
  - **InnerHtml** - obsahuje celý ”tag”, element i spolu s vnitřním textem.

  - **XAttributes** - obsahuje kolekci(HtmlAttributeCollection) všech atributů obsažených v daném elementu.
  - **GetAttVal**(*string*) - funkce vrací hodnotu předaného atributu v konstruktoru.

- **Table** - Pomocná třída určená pro rychlé a jednoduché vytváření složitějších databázových tabulek.
  Před použitím těchto metod je potřeba nejdříve zavolat TableName(string) viz. funkce createTable() ve zdrojovém kódu č. 9.

  - **AddAttribute**(*string*) - přidá atribut do tabulky, a vrátí objekt typu Table.
  - **AddAttribute**(*string,string,int*) - stejná metoda pouze má v konstruktoru definované parametry. Například (”’nadpis”,”VARCHAR”,80).
  - **AddPrimAttribute**(*string*) - stejná metoda jako AddAttribute(*string*), s rozdílem že přidaný atribut bude primární klíč.
  - **AddPrimAttribute**(*string,string,int*) - stejně tak s přidanými parametry jako AddAttribute(*string,string,int*).

4.2.3 Globální proměnné

V našem případě slouží globální proměnné k definování názvu výstupní databáze a její tabulky. Tyto parametry jsou pak interně předány metodě **Save**(*params object[]*) popsané v kapitole 4.2.1 . Parametry musí být vždy definovány na začátku každého skriptu.

- **DATABASE** – určuje název databázového souboru, do kterého chceme vkládat výsledky.
- **TABLE_NAME** – nastavuje název tabulky v databázi, do které se budou data zapisovat.

4.2.4 Struktura skriptu

Struktura skriptu je rozdělena na 2 části, pomocí dvou hlavních funkcí:

- **createTable**() - tato funkce se spouští pouze v případě, pokud není ještě pro skript vytvořené databázové místo. Nebo tabulka, kterou je potřeba vytvořit.

- **update**() - tato část skriptu se pouští bezprostředně po vytvoření tabulky, při každém spuštění skriptu s již vytvořenou tabulkou.

4.3 Ukázka skriptu

```
DATABASE = "test.sdb"
TABLE_NAME = "jakpsatweb"


function update()
  start = HttpGet("http://www.jakpsatweb.cz/")
  links = GetAllPagesUnder(start)

  for x in enum(links) do
    page = HttpGet(links[x].AbsoluteUri)
    nadpis = page:Xpath("//h1")
    for y in enum(nadpis) do
      data = {"nadpis",nadpis[y].InnerText,"text",links[x].AbsoluteUri}
      Save(data)
      break
    end
   end
  Transaction()
end
function createTable()
   ct = TableName(TABLE_NAME):AddPrimAttribute("nadpis","VARCHAR",255):
       AddAttribute("text","VARCHAR",255)
   AddTable(ct)
end
```

Výpis 9: Ukázka použití vlastního API ve skriptu

5 Úložiště dat

5.1 Co je to SQL

SQL je dotazovací jazyk, který je používán při práci s daty nad relačními databázemi. Vývoj jazyka SQL začal někdy v 70. letech, kdy se původně jazyk nazýval SEQUEL. Později byl však přejmenován na nynější název a to SQL.V 80. letech bylo SQL standardizováno jako SQL-86 podle roku 1986.

Po uplynutí dalších 3 let byl jazyk vylepšen a uveden jako SQL-89. V roce 1992 byla provedena první úprava tohoto ISO standardu, který je známý pod SQL2 nebo SQL-92. Aktuální verzí je SQL3 neboli SQL-99.

5.2 Dotazy v SQL

Dotazování v SQL se dělí podle využití do pomyslných 4 kategorií:

- Příkazy pro manipulaci s daty (SELECT, INSERT, UPDATE, DELETE, . . .)
- Příkazy pro definici dat (CREATE, ALTER, DROP, . . .)
- Příkazy pro řízení přístupových práv (GRANT, REVOKE)
- Příkazy pro řízení transakcí (START TRANSACTION, COMMIT, ROLLBACK)

5.3 SQLite

SQLite je relační databázový systém, který je volně šířený pod licencí public domain, neboli volné dílo.

Celý SQLite je jedna malá knihovna, kterou připojíme k aplikaci. Pomocí jednoduchého rozhraní ji lze ihned začít používat. Další zajímavou vlastností SQLite je přenositelnost souborů mezi různými operačními systémy.

5.4 Ukázka SQLite v C#

Na stránkách `http://www.sqlite.org/`, si stáhneme soubory, které potřebujeme. Pro .NET aplikace je najdeme v sekci **Precompiled Binaries For .NET**. Soubory jsou tříděné ve

složkách podle architektury procesoru (*x86/x64*). Dále dle použitého .NET Frameworku. V našem případě tedy Framework 4.0.

Především nás zajímají soubory:

- System.Data.SQLite.dll
- SQLite.Interop.dll

Po naimportování těchto knihoven do projektu můžeme pracovat s SQL databázemi.

```
using System.Data.SQLite;
...
string connectionString = string.Format("Data Source=DB.sdb");
string command = "INSERT INTO ...";

var cnn = new SQLiteConnection(connectionString);
cnn.Open();
var mycommand = new SQLiteCommand(cnn);
mycommand.CommandText = command;
int rowsUpdated = mycommand.ExecuteNonQuery();
cnn.Close();
```

Výpis 10: Propojení C# a SQLite

6 Implementace Web crawleru

V této části už máme vybrán skriptovací jazyk a úložiště pro výsledná data. Můžeme přejít k části věnující se samotné implementaci.

K implementaci jsem použil nástroj MS Visual Studio 2010, který umožňuje programovaní v jazyku C#, jenž je nutnou součástí této práce. Samotnou aplikaci jsem vytvořil tak, abych zachoval základní koncepci oddělení logické části od grafického rozhraní.

6.1 Naprogramování crawleru

Jeden z nejdůležitějších kroků bylo sestavení samotného robota pro stahování webových stránek. Koncept pracuje na principu načtení webové stránky z které postupně extrahuje URL adresy.

Ve výsledku bylo třeba ošetřit několik vzniklých situací. Vezměme si například fakt, že místo obvyklého odkazu na webový obsah, byla použita emailová adresa ve standardním HTML formátu (*mailto:emailová@adresa*). Takováto adresa by neprošla přes parser, jelikož se nejedná o webovou stránku. Tento typ adres je potřeba předem ignorovat. Zároveň musíme počítat s problémem starých neboli nefunkčních odkazů, se kterými nelze dále pracovat.

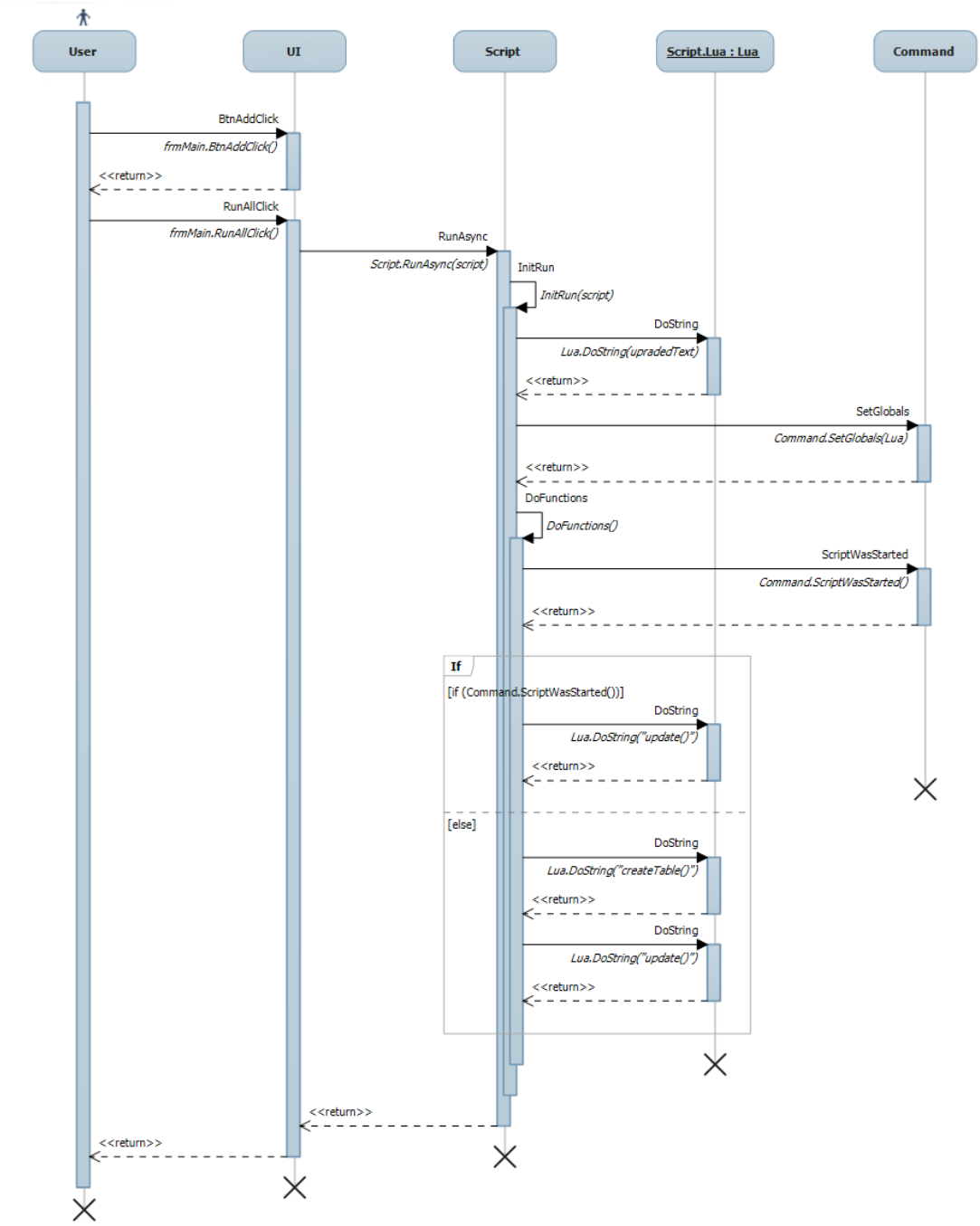
6.2 Spouštění skriptu

Nyní popíšu příklad spuštění skriptu uživatelem. Pomocí tlačítka “Přidat skript“ jej přidá do seznamu, kde se zobrazí ve stavu “Čeká na spuštění“. Následně klikne na tlačítko “Spustit“ a skript se začne zpracovávat.

Protože nechceme, aby nám při zpracovávání přestal reagovat [2] celý program, spouští se asynchronní metoda RunAsync(). Ta dále obalí skript potřebnými údaji pro správny chod (např. funkce enum(), kterou Lua v sobě nemá zakomponovánu). Poté se spustí DoString() s takto upraveným textem. Hlavnímu programu se předají globální proměnné viz. 4.2.3. Posledním krokem ověřujeme, zda v minulosti uživatel tento skript spustil, pokud ne spustíme pomocí Lua.DoString navíc metodu CreateTable() před metodou update().

[2] Může nastat situace, kdy hlavní vlákno zatížime natolik, že není schopno přijímat další přikazy uživatele (např. obsluhu programu), a tudíž se program jevý jako zaneprázdněný (tzv. zamrznutí).

Nyní již záleží na daném skriptu, co vše bude z hlavního programu spouštět.



Obrázek 1: Sekvenční diagram spouštění skriptu uživatelem

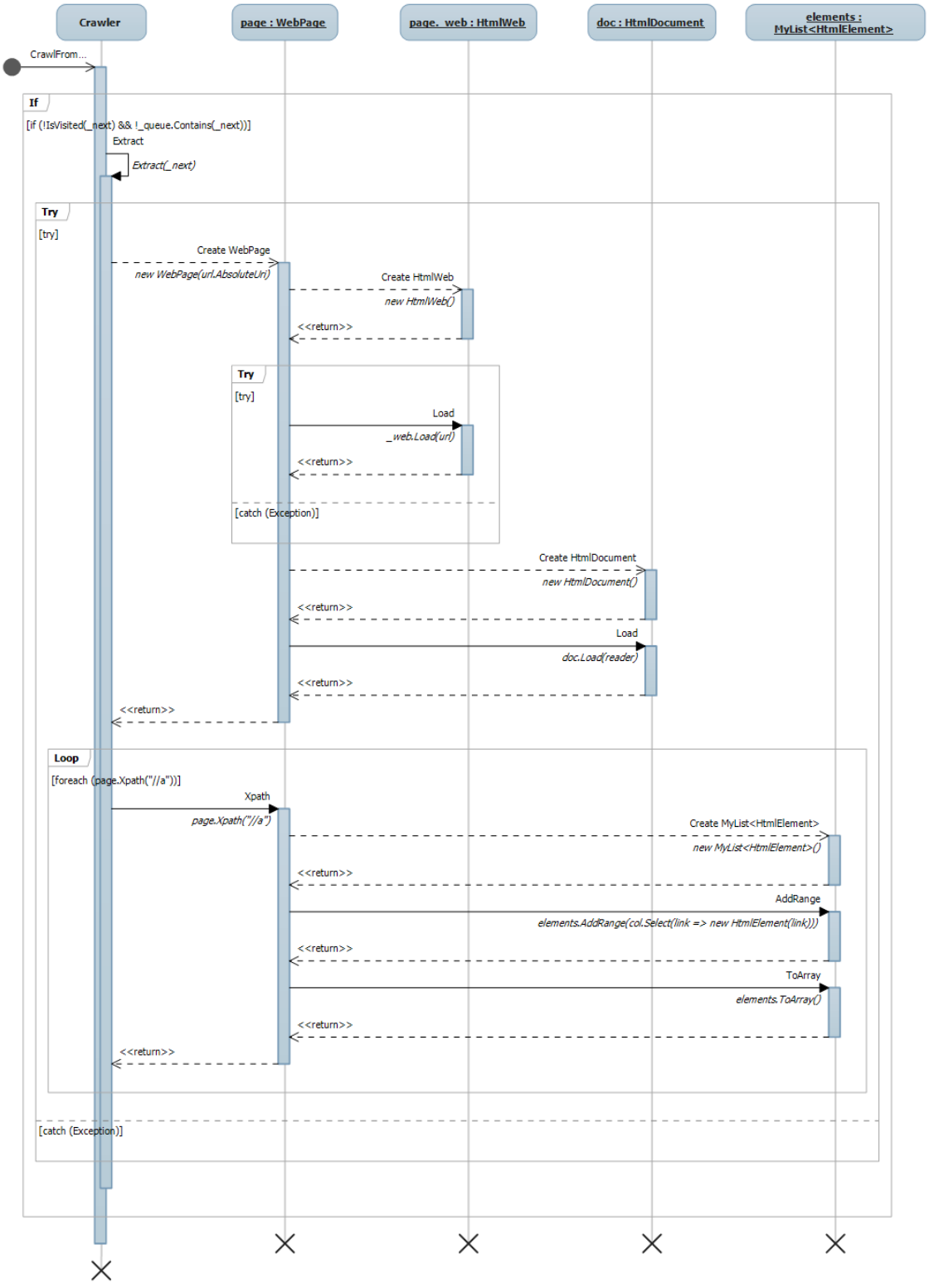
6.3 Metoda GetAllPagesUnder

V této sekci nastíním popis hlavní funkce crawleru, která prohledává webový obsah. Metoda nepřijímá žádný parametr a vrací false pokud je fronta s odkazy prázdná.

Na začátku metoda zkontroluje, jestli již danou adresu nenavštívila [3] . Pokud byla tato adresa již navštívena, vybere jinou z fronty odkazu. Dále proběhne extrahování odkazů. První se daná url adresa předá třídě WebPage, ta se postupně postará o načtení webové stránky pomocí tříd HtmlWeb a HtmlDocument, obsažené v knihovně HTMLAgitilyPack. Ty nám předají důležité parametry, jako je například InnerHTML [4] atd. Jakmile máme všechny potřebné údaje o stránce, můžeme spustit vyhledávání odkazů ze zdrojového kódu. Odkazy vyhledáváme pomocí Xpath dotazů, jednoduchým dotazem "//a". Proces vybere z tagu "a" [5] hodnotu atributu "href". Obsah tohoto atributu přidá do fronty. Před samotným přidáváním ještě vyřazujeme odkazy v nesprávném formátu. Zjednodušený postup je prezentován na sekvenčním diagramu (viz strana 26).

Tato metoda se spouští v Lua skriptu pomocí příkazu GetAllPagesUnder (viz 4.2.1).

[3] Pomocí metody IsVisited().
[4] Zdrojový kód stránky.
[5] <a href=" "/>



Obrázek 2: Sekvenční diagram metody CrawlFromQueue

6.4 Vlastní DLL knihovna

Veškerá logika aplikace (propojovaní ke skriptům, sql, crawler funkce atd.) je implementována jako samostatná knihovna. Tuto knihovnu jsem pojmenoval jako clib.dll, což je zkratka pro crawler library.

Důležité třídy obsažené v knihovně :

- **Command.cs** - slouží k registraci funkcí do Lua. Také obsahuje všechny funkce, které můžou být spouštěny ze skriptu.
- **Crawler.cs** - logická část crawleru, stahuje stránky, řadí do fronty.
- **Script.cs** - slouží ke spouštění skriptů.
- **Sql.cs** - obsluha SQL databází , umožňuje vkládaní, vytváření, selekci a další operace nad databází.
- **Table.cs** - pomocná třída pro tvorbu tabulek. Umožňuje vytváření tabulkových příkazů bez pamatování si pravidel pro vyvaření tabulek. Omezena pouze na jednoduché tabulky bez cizích klíčů atd.
- **WebPage.cs** - třída, která slouží jako HTML parser.

6.5 Ostatní použité knihovny

Kromě knihoven uvedených pro obsluhu skriptovacího jazyka a SQL databáze, nalezneme v aplikaci ještě další dvě knihovny:

- **HtmlAgilityPack.dll** - HtmlAgilityPack je HTML parser, který je postaven na DOM. Podporuje dotazy Xpath, jenž umožňují parsování HTML stránek.
- **ScintillaNET** [6] - je komponenta pro Windows Form aplikace, sloužící jako textový editor. Mezi největší přednosti patří zvýrazňování syntaxe, a automatické dokončování kódu. Pro jazyky HTML, Python, XML je již zvýrazňovaní definováno podle standardních editorů.

[6] Pro správnou funkčnost editoru, je potřeba vložit do složky Windows/System32 knihovnu SciLexer.dll (pro 64bit knihovnu SciLexer64.dll do složky Windows/SysWOW64). Tyto knihovny naleznete v příloze.

7 Obsluha aplikace WebCrawler+

Aplikaci spustíme volbou: **Přidat skript** – **Prohledat**, vybereme skript, např. z příloh na CD ve složce ″Skripty″. Stiskneme tlačítko **Přidat skript**. Skript se automaticky přidá do fronty ve stavu *″Čeká na spuštění″*.

Nyní zvolíme **Spustit** – **Vše**. V dolní liště vidíme průběh zpracovaní skriptu v procentech. Konec skriptu indikuje stav *″Dokončen″*. Nyní můžeme zvolit **Prohlížet** – **Zobrazit aktuálně zvolený skript**.

V prohlížeči vidíme seznam všech výsledků, zpracovaných skriptem. Ty můžeme dvojklikem zobrazit detailněji a označovat jako již přečtené.

Aplikace byla testována a vykazuje plnou funkčnost pod Windows 8 a Windows 7 s .NET Frameworkem 4.0.

8 Použití aplikace WebCrawler+

Pro nastínění použití této aplikace jsem napsal 2 skripty:

1. Stahuje informace z blogu prof. Ing. Ivo Vondráka, CSc. rektora VŠB.

```
DATABASE = "rektor.sdb"
TABLE_NAME = "blog"

function update()
  start = HttpGet("http://www.vsb.cz/blog/cs/von05/")
  links = GetAllLocalPagesUnder(start)

  for x in enum(links) do
    page = HttpGet(links[x].AbsoluteUri)
    nadpis = page:Xpath("//div[@class=\"blog-item\"]//h2")
    text = page:Xpath("//div[@class=\"content\"]")
    for y in enum(nadpis) do
     h2 = nadpis[y].InnerText
       for muj in enum(text) do
         txt = text[muj].InnerText
         break
       end
     data = {"nadpis",h2,"text",txt,"zdroj",links[x].AbsoluteUri}
     break
    end
    Save(data)
  end
  Transaction()
end

function createTable()
  ct = TableName(TABLE_NAME):AddPrimAttribute("nadpis","VARCHAR",255):
      AddAttribute("text","VARCHAR",200000):AddAttribute("zdroj","VARCHAR"
      ,255)
  AddTable(ct)
end
```

Výpis 11: Ukázka skriptu č.1

2. Stahuje obrázky map areálů VŠB.

```
DATABASE = "mapaArealu.sdb"
TABLE_NAME = "local"

function update()
  start = HttpGet("http://www1.vsb.cz/arealy/cz/")
  links = GetAllLocalPagesUnder(start)

  for x in enum(links) do
    page = HttpGet(links[x].AbsoluteUri)
    obrazek = page:Xpath("//img")
    for y in enum(obrazek) do
      if(obrazek[y]:GetAttVal("src")== null)then
      else
        combine = CombinePath(links[x].AbsoluteUri,obrazek[y]:GetAttVal("src
            "))
        data = {"obrazek",combine,"popis",obrazek[y]:GetAttVal("alt"),"url",
            links[x].AbsoluteUri}
        Save(data)
      end
    end
  end
  Transaction()
end

function createTable()
  ct = TableName(TABLE_NAME):AddPrimAttribute("obrazek","VARCHAR",255):
      AddAttribute("popis","VARCHAR",255):AddAttribute("url","VARCHAR",255)
  AddTable(ct)
end
```

Výpis 12: Ukázka skriptu č.2

9 Závěr

Vývoj webových crawlerů a samotné vyhledávání mě zajímalo již dříve. Nikdy jsem se však touto problematikou nezaobíral do detailů, jako při tvorbě této práce.

Rozšířil jsem si znalosti z oblasti webových vyhledávačů, *DOM* parserů a jejich možné využití v praxi. Za největší přínos považuji získané znalosti o integraci skriptovacích jazyků do *C#*. Jelikož jsem se s tímto předtím nesetkal, považoval jsem toto téma částečně i jako výzvu. Skriptovací jazyky mi otevřeli možnosti, jak vytvářet aplikace, které by mohli být dále rozšiřovány uživateli, za pomocí vlastních skriptů. Tím se může stát aplikace poměrně multifunkční. Tyto znalosti se mi určitě budou hodit do dalšího studia a později i do profesního života.

Do budoucna mě napadá spousta vylepšění jako je například zrychlení crawleru pomocí více vláken. Dále bych chtěl zlepšit způsob zobrazování výsledných dat, přidat nové metody propojené se skripty pro širší možnosti využití a mnohé další.

Myslím si, že jsem všechny požadavky této práce splnil, narazil jsem na množství problémů, které se mi podařilo úspěšně vyřešit. Tato práce mi dala mnoho poznatků, a proto ji hodnotím jako velmi přínosnou.

Jakub Gereg

10 Reference

[1] IERUSALIMSCHY, Roberto. *Programming in Lua.* 2nd ed. Rio de Janeiro: Lua.org, c2006, xvii, 307 s. ISBN 85-903-7982-5.

[2] WU, Chaur. *Pro DLR in .NET 4.* New York: Distributed to the book trade worldwide by Springer Science Business Media, c2010, xviii, 305 p. Expert's voice in .NET. ISBN 14-302-3066-5.

[3] Python. *Wikipedie, otevřená encyklopedie* [online]. 12. 4. 2012 [cit. 2012-05-04]. Dostupné z: http://cs.wikipedia.org/wiki/Python

[4] IronPython. *CodePlex - Open Source Project Hosting* [online]. 23 Mar 2012 [cit. 2012-05-04]. Dostupné z: http://ironpython.codeplex.com/

[5] Web crawler. *Wikipedia, the free encyclopedia* [online]. 3 May 2012 [cit. 2012-05-04]. Dostupné z: http://en.wikipedia.org/wiki/Web_crawler

[6] About. *The Programming Language Lua* [online]. Tue Apr 17 18:42:43 BRT 2012 [cit. 2012-05-04]. Dostupné z: http://www.lua.org/about.html

[7] Luainterface. *Luainterface - A library for integration between the Lua language and Microsoft .NET platform's Common Language Runtime (CLR)* [online]. Feb 2010 [cit. 2012-05-04]. Dostupné z: http://code.google.com/p/luainterface/

[8] DynamicLua. *CodePlex - Open Source Project Hosting* [online]. 11. vyd. 7 Feb 2012 [cit. 2012-05-04]. Dostupné z: http://dynamiclua.codeplex.com/

[9] Normalized Comparison. *Programming Language Popularity* [online]. 13 Apr 2012 [cit. 2012-05-04]. Dostupné z: http://langpop.com/

[10] Javascript .NET. *CodePlex - Open Source Project Hosting* [online]. 26 Apr 2012 [cit. 2012-05-04]. Dostupné z: http://javascriptdotnet.codeplex.com/

[11] Co je Google V8 JavaScript Engine? *GUG.cz: Česká Google User Group* [online]. 3. září 2008 [cit. 2012-05-04]. Dostupné z: http://clanky.gug.cz/2008/09/co-je-google-v8-javascript-engine.html

[12] V8 JavaScript Engine. *Google Code* [online]. Dec 2008 [cit. 2012-05-04]. Dostupné z: http://code.google.com/p/v8/

[13] IronRuby. *CodePlex - Open Source Project Hosting* [online]. 14 Mar 2011 [cit. 2012-05-04]. Dostupné z: http://ironruby.codeplex.com/

[14] Skriptování v .NETu. *Vyvojar.cz - Vývojáři sobě* [online]. 07 October 2009 [cit. 2012-05-04]. Dostupné z: http://blog.vyvojar.cz/topas/default.aspx?p=2

[15] Html Agility Pack. *CodePlex - Open Source Project Hosting* [online]. 18 Jan 2011 [cit. 2012-05-04]. Dostupné z: http://htmlagilitypack.codeplex.com/

[16] SQL. *Wikipedie, otevřená encyklopedie* [online]. 1. 5. 2012 [cit. 2012-05-04]. Dostupné z: http://cs.wikipedia.org/wiki/SQL

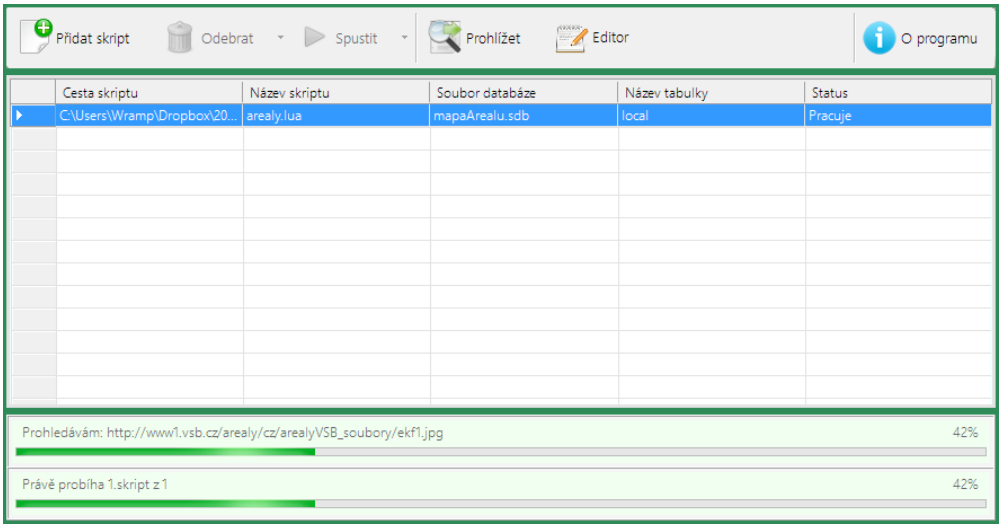
A Uživatelské rozhraní

V uživatelském rozhraní jsem se snažil vytvořit příjemné a vyhovující prostředí pro uživatele.

A.1 Aplikace

Celá aplikace se rozděluje na:

- **Horní panel** - obsahuje důležité prvky pro obsluhu programu.
- **Fronta skriptů** - zobrazuje stavy jednotlivých skriptů.
- **Dolní panel** - průběžně informuje uživatele o stavu prováděného skriptu.



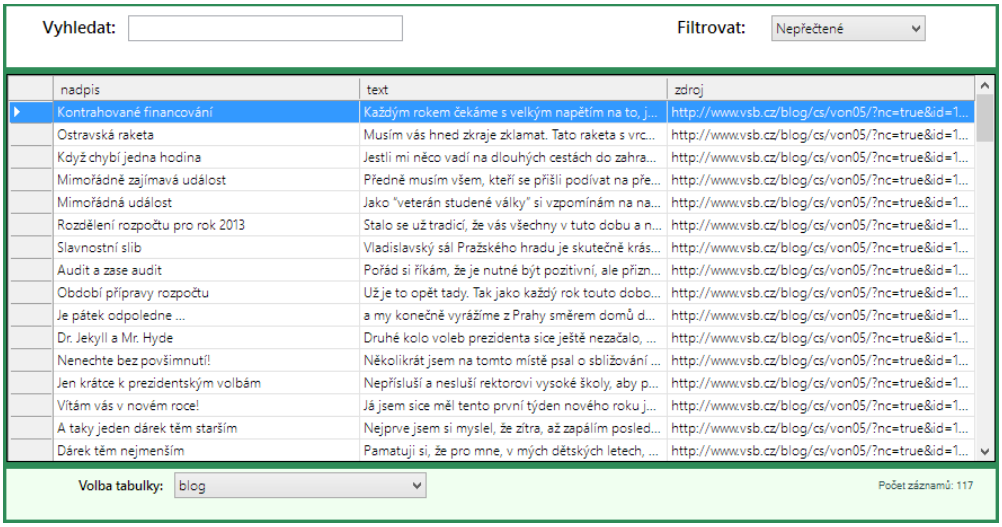
Obrázek 3: Hlavní stránka aplikace - zobrazení průběhu

A.2 Prohlížení výsledných dat

Prohlížeč se skládá z:

- **Vyhledávání** - usnadňuje práci s výsledky
- **Filtr** - umožňuje filtraci mezi přečtenými/nepřečtenými daty.

- **Tabulka** - zobrazuje data.
- **Dolní panel** - slouží pro výběr jednotlivých sloupců v tabulce.

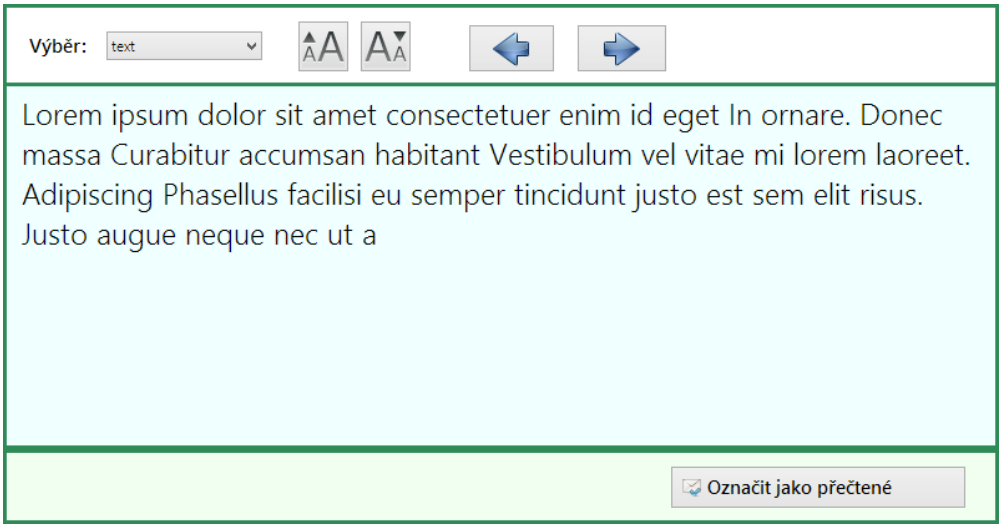


Obrázek 4: Vzhled prohlížeče

A.3 Detailní zobrazení

Skládá se z:

- **Výběru** - vybíráme sloupec, který chceme zobrazit.
- **Šipek pro procházení ostatních výsledků** - zobrazuje předchozí/následující výsledky.
- **Nastavení velikosti písma**
- **Tlačítka pro označení přečtěného článku**



Obrázek 5: Detail výsledků

A.4 Ukázka editoru

Editor se skládá z:

- **Horní panel** - umožňuje vytvářet, otevírat a ukládat.
- **Levý panel** - zobrazuje čísla řádků.

```
1  DATABASE = "test.sdb"
2  TABLE_NAME = "jakpsatweb"
3
4
5  function update()
6      start = HttpGet("http://www.jakpsatweb.cz/")
7      links = GetAllPagesUnder(start)
8
9      for x in enum(links) do
10         page = HttpGet(links[x].AbsoluteUri)
11         nadpis = page:Xpath("//h1")
12         for y in enum(nadpis) do
13             data = {"nadpis",nadpis[y].InnerText,"text",links[x].AbsoluteUri}
14             Save(data)
15             break
16         end
17     end
18      Transaction()
19 end
20 function createTable()
21     ct = TableName(TABLE_NAME):AddPrimAttribute("nadpis","VARCHAR",255):AddAttrik
22     AddTable(ct)
```

Obrázek 6: Ukázka vzhledu editoru

B Obsah na CD

CD obsahuje tyto složky:

- **BPText** - text k bakalářské práci ve formátu PDF.
- **Ke spuštění** - všechny potřebné soubory ke spuštění aplikace.
- **Obrázky** - screenshoty uživatelského rozhraní a ostatní obrázky.
- **Skripty** - skripty pro práci s programem.
- **VSProjekt** - Visual Studio projekt.